ISSN 1349-1121
JAXA-RM-09-002

宇宙航空研究開発機構研究開発資料

JAXA Research and Development Memorandum

非構造格子NS ソルバによるJAXA 統合スパコン（JSS）の性能評価

坂下　雅秀，松尾　裕一，村山　光宏

2009 年 7 月

宇宙航空研究開発機構

Japan Aerospace Exploration Agency

非構造格子 NS ソルバによる JAXA 統合スパコン(JSS)の性能評価*

坂下　雅秀*1,　松尾　裕一*1,　村山　光宏*2

Performance evaluation of the JAXA new supercomputer system JSS by an unstructured NS solver

Masahide Sakashita*1 and Yuichi Matsuo*1 and Mitsuhiro Murayama*2

Abstract

Japan Aerospace Exploration Agency (JAXA) has installed a new 100-terascale massively parallel supercomputer system as part of the JAXA Supercomputer System (JSS) for aerospace science and engineering research purposes. It is in full operation from April 2009. In this study, we investigate and evaluate the performance of CPU operation and memory access of the system by using an unstructured NS solver, and found that the new system has an expected high performance compared with the previous system. However, we also found that for high efficiency to applications, the issues on load unbalance and recursive reference can be further considered.

Key Word: Supercomputing, OpenMP, MPI, Navier-Stokes, CFD, unstructured mesh

概　要

本稿では，2009 年 4 月より稼動した JAXA 統合スーパーコンピュータのうち，大規模並列計算機システムにおいて，非構造格子 NS ソルバを評価用コードとして，主に演算性能とメモリアクセス性能の評価を行い，システム特性やマルチコア CPU の使い勝手について調べるとともに，利用指針や技術課題について考察した結果を報告する．新システムは，マルチコアを有効利用するために IMPACT と呼ばれる機能を有するが，IMPACT 並列実行により，旧システムと比較して十分高い性能を実現可能であることが確認できた．一方で，より効率的な利用を考えた場合には，負荷バランスや再帰参照などの扱いを考慮した方が良いこともわかった．

* 平成 21 年 06 月 22 日受付
*1 情報・計算工学センター　計算機運用・利用技術チーム
（JAXA's Engineering Digital Innovetion Center, Computing Resource Management Team）
*2 航空プログラム　国産旅客機チーム
（Aviation Program Group, Civil Transport Team）

1. はじめに

JAXA では，旧宇宙 3 機関（宇宙開発事業団，宇宙科学研究所および航空宇宙技術研究所）時代からの経緯で，調布事業所及び角田事業所，相模原キャンパスの 3 ヵ所にスーパーコンピュータ（以下，スパコンと略す）を保有して来たが，調布事業所と角田事業所のスパコンがほぼ同時期にリースアウトするのを機に，JAXA 統合スーパーコンピュータ（JAXA Supercomputer System，JSS）として調達手続きを実施した．2009 年度当初からの本稼動を目指して導入が進められ，一部については 2008 年 4 月より暫定的な運用が開始された. JAXA 統合スパコンは，スパコンによる数値シミュレーション技術を宇宙開発等の JAXA 事業に本格的に活用することを企図し，一方では，宇宙 3 機関統合のシンボル的な位置づけとして導入されたものである．

従来のスパコンでは，ノード[1]あたりのメモリ量を多くしたい等の理由で，100 台規模の CPU が対称的にメモリを共有する大規模 SMP（Symmetric Multi Processor）システムをノードとし，高々100 程度のノードをネットワークで結合するタイプ（SMP クラスタ）が多く，調布事業所においても採用されていた．しかし，SMP クラスタシステムにおいては，流体計算を行うにはメモリアクセス性能が低くそれが性能ネックになる，あるジョブが他のジョブの影響を受けて計算時間がばらつき運用性に支障が出る，等の問題が顕在化した．一方，最近の半導体技術の進展を受けて，CPU チップ内に複数のプロセッサ・コアを内蔵した「マルチコア」と呼ばれるアーキテクチャが主流となり，これをどのように有効に使うかが技術的課題となっている．これを受けて JSS では，これらの問題に対処するため，後述するマルチコアを有効に使う IMPACT と呼ばれる機能を搭載し，ノードあたり 1CPU と極めて小さな規模のノードを多数（1000 ノード以上）結合したスカラー超並列タイプのシステムを導入した．こうした構成のスパコンは世界的にも例がなく，JAXA にとっても初の経験のため，JAXA の保有する流体コードにおける処理性能や利用方法には未知な部分が多い．

そこで，JAXA の航空分野における主要な流体コードの一つである非構造格子 NS（Navier-Stokes）ソルバを用いて，JSS のスカラー超並列システムの処理性能の評価を行うとともに，マルチコアシステムの特性を分析し，利用指針や課題の考察を行った．本稿では，その結果を報告する．

2. JSS の概要

JAXA 統合スパコン（JSS）は，図 2.1 に示すように，スカラー大規模並列計算機システム，ストレージシステム，共有メモリサブシステム，ローカルサーバ等より成る．このうち．スカラー大規模並列計算機システムは，富士通製の FX1 と呼ばれる小規模ノードを多数，ネットワークで結合させたスカラー超並列分散メモリシステムである．120TFLOPS[2]の処理性能，94TB のメモリを有するメインシステムと，15TFLOPS，6TB のプロジェクトシステムから成る．120TFLOPS のメインシステムは 3,008 ノード，15TFLOPS のプロジェクトシステムは 384 ノードから構成される．各ノードはクロック周波数 2.5GHz，マルチコア（4 コア）の SPARC64 VII チップを 1 個搭載し，40GFLOPS の処理性能と 32GB のメモリ容量，40GB/秒のメモリアクセス性能を有する．ノード内の 4 つのコアは，2 次キャッシュおよびメインメモリを共有しており，さらにコア間のハードバリア機構やコンパイラ技術と合わせて，主に IMPACT (Integrated Multi-core Parallel Architecture)と呼ばれる機能を用いた自動並列化による利用が想定されている．IMPACT とは富士通独自の翻訳機能であり，CPU チップ内のコア数を含めた利用可能資源が固定であることを考慮したマルチコア向けの自動スレッド並列化オプションである．ノード間は InfiniBand[*1*]によりフルバンド幅のファットツリーと呼ばれるトポロジーで結合され，MPI[*2*]に代表されるプロセス並列化による利用が想定されている．ノード間に対しては，バリア同期や集合通信を高速化するために，バリアネットワークと呼ばれるネットワークが主ネットワークとは別に設けられ，ノード間バリアや集合通信の高速化が図られている．

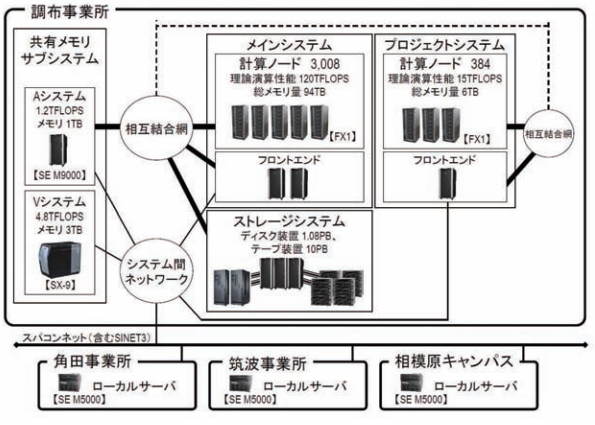


図 2.1 JAXA 統合スパコンの構成

本報告の性能評価には，このうちのプロジェクトシステム（384 ノード，図 2.2）を使用した．プロジェクトシステムにおける MPI プログラムの並列実行モデルとしては，IMPACT 機能を用いたノード内自動スレッド並列（4 並列）とノード間プロセス並列を組み合わせたハイブリッド並列実行モデル（図 2.3(a)），または IMPACT

[1] スパコンは，通常の場合，単位計算機がネットワークで結合された形態をとる．この単位計算機のことを「ノード」と呼ぶ．

[2] 1TFLOPS は浮動小数点演算を 1 秒間に 1 兆回実行する能力．

並列を用いず MPI プロセス並列だけを用いる純 MPI 実行モデル（図 2.3(b)）を採用している．IMPACT 並列実行は、もとになる MPI 並列プログラムに対して，コンパイル時にオプションを指定するのみで実行することができる．ただし，使用するコアの数は純 MPI モデルの 4 倍必要となる．また，ノード内では IMPACT 並列に加えて，ディレクティブの挿入とコンパイル・オプションの指定により OpenMP[3]によるスレッド並列実行も可能である．

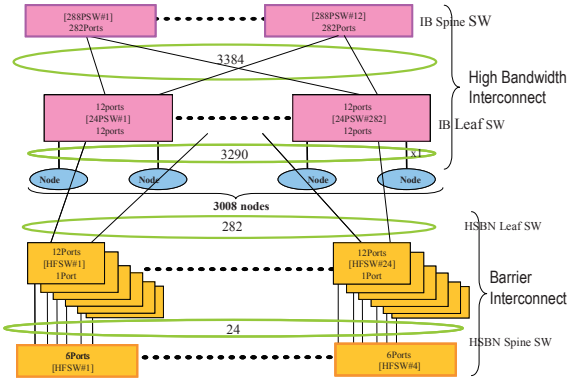
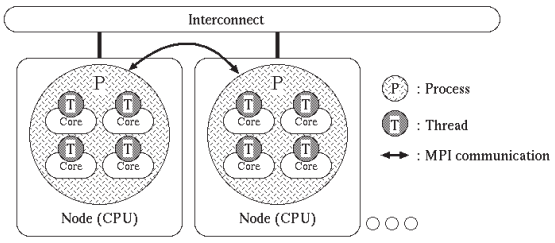
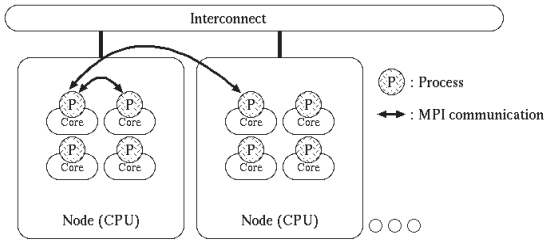


図 2.2　プロジェクトシステムの構成



(a) IMPACT 実行モデル



(b) 純 MPI 実行モデル

図 2.3　プロジェクトシステムにおける並列実行モデル

3.　性能評価コード概要

本報告の性能評価に使用したコードでは，3 次元 NS 方程式を空間方向にはセル節点有限体積法により，時間方向には Euler 陰解法により離散化している．空間流束の評価は HLLEW 法[4]により行い，高次精度差分における制限関数として Venkatakrishnan の制限関数[5]を用い，時間積分には LU-SGS 陰解法[6][7]を用いている．空間格子として非構造格子法を用いており，構造格子に比べて格子生成が比較的容易で，流れ場の重要な場所に格子を細分化し，局所的重点的配置することにより精度向上を図ることが可能である．一方で，格子のならびに規則性がないため，間接的あるいは不規則なメモリアクセスが多くなり，アクセス効率において不利になる．

並列化は，MPI を利用したプロセス並列により行っている．プロセス並列化を行うにあたっては，まず計算に先立って各プロセスに割り当てるために格子空間を領域分割し，この分割された領域をそれぞれのプロセスが分担して計算する[8]．そして，非構造格子に適用可能な超平面（Hyper Plane）を構成することにより，各プロセスに割り当てられた領域に LU-SGS 法を適用する．

有限体積法においては，流束ベクトルは，各検査体積を構成する面ごとに求める必要があるが，ある面は異なる二つの検査体積の境界を構成するので，その面を通る流束はその二つの検査体積に対しては同じ量でかつ符号が反対であるように寄与する．従って，流束ベクトルを検査体積ごとにそれを構成する全ての面に対して計算することは，流束ベクトルを 2 回計算することとなり効率的ではない．そこで，面ごとに計算すれば流束ベクトルの計算は 1 回で済む．セル節点体積法の場合，検査体積がその唯一含む節点の番号で指定されるのと同様に，面はそれが唯一含む辺の番号で指定されるので，実際のプログラムにおける流束の計算は，（辺 IE の両端の節点番号 N1，N2 を保持する配列の名前を"NEDG2ND"とすると）リスト 3.1 のようになる．ここで，この DO ループは配列"FLUX"に対して再帰参照になっていることに注意されたい．なぜなら、検査体積は複数の面から構成されているため，異なる面（辺）IE において同じ検査体積番号（節点番号）が N1 または N2 に現れるからである．このようなコーディング方法でプログラムされたコードは，ベクトル化やスレッド並列化ができないことから，ここではこれをスカラ版と呼ぶことにする．

リスト 3.1　流束計算の例

```
    DO 100 IE = 1, Num_edges
       N1=NEDG2ND(1,IE)
       N2=NEDG2ND(2,IE)
       HLLEW=FLUX_FUNC(arguments)
       FLUX(N1)=FLUX(N1)+HLLEW
       FLUX(N2)=FLUX(N2)-HELLW
100 CONTINUE
```

このような再帰参照は，色分け（Coloring）によるグループ化の手法を用いることにより回避することができ，ベクトル化やスレッド並列化が可能となる[9]．これは，ある検査体積（節点）に含まれる全ての面（辺）を，必ず別の色を持つように前もってグルーピング（色分け）しておき，DOループ内では同じ色を持った面（辺）のみを計算することにより，再帰参照を回避する．この場合，実際のコーディングは，例えばリスト 3.2 に示すような二重ループになる．外側のDOループが全ての色に対する処理のループであり，内側のDOループがその内

のある一つの色に対する処理を行うDOループである．色分けを適切に行うことにより，内側のループで一度に処理される面（辺）は必ず異なる検査体積（節点）を構成するものとなり，再帰参照が回避されベクトル化やスレッド並列化が可能となる．このようなコーディング方法でプログラムされたコードを，ここではスレッド版と呼ぶことにする．

リスト 3.2 流束計算のスレッド並列化例

```
      DO 100 IC = 1, MAX_Colors
*VOCL LOOP,NOVREC
      DO 110 IE = 1, Num_edges(IC)
         N1=NEDG2ND(1,IE,IC)
         N2=NEDG2ND(2,IE,IC)
         HLLEW=FLUX_FUNC(arguments)
         FLUX(N1)=FLUX(N1)+HLLEW
         FLUX(N2)=FLUX(N2)-HELLW
  110 CONTINUE
  100 CONTINUE
```

スレッド版では，さらに並列実行時の性能向上を目的として、以下の二点が考慮している．

(1) LU－SGS 法における節点番号の付け替え

(2) 色分けの削除及び DO ループの分割

この二点についての具体的な内容を以下に示す．

● LU-SGS 法における節点番号の付け替え

性能評価コードでは，非構造格子に LU-SGS 法を適用するために超平面が構成されている．ところが，節点における各物理量等を配列に保存する際には，格子生成時に付されたオリジナルの節点の番号でインデックスされる場所に保存されている．このような方法は，あるひとつの超平面内に存在する節点の番号が不連続であるため，効率的なメモリアクセスを疎外する要因となる．そこで，LU-SGS 法の計算を行う部分では，超平面を考慮した節点番号の付け替えを行い，超平面内で節点番号が連続になるようにしている．

● 色分けの削除及び DO ループの分割

スレッド並列化では，スレッド生成のオーバーヘッドをなるべく小さくするために，スレッド生成回数の少ない，より外側の DO ループで並列化することが望ましい．色分けによるスレッド並列化では，リスト 3.2 において内側の DO ループである DO 110 が並列化されることとなり，スレッド生成のオーバーヘッドによる性能低下が予想される．加えて，スレッド並列化される DO ループの回転数は，生成されたスレッドの中でなるべく多くの演算が行われるように，なるべく多くなることが望ましいが，色分けによるスレッド並列化では、一度に計算されるのは一つの色に属する辺のみであり，全ての辺を一度に処理するのに比べて性能劣化が予想される．一方で，全ての辺について同時に計算することにすれば，リスト 3.1 に示すように DO ループは一重ループとなり，外側かつ回転数の多い理想的なループの構成となるが，再帰参照を含むため，スレッド並列化を行うことが出来ない．

このため、リスト 3.3 に示すように色分けを削除するとともに，DO ループの分割を行うことで，色分けによる問題の改善を図る．リスト 3.3 では，DO 100 において辺ごとに計算された流束ベクトルが，一旦，作業用配列"EDG_WK"に辺ごとの値として保存された後，DO 110 において，節点（検査体積）ごとの配列"FLUX"に足し込まれている．ここで、DO 100 における配列"EDG_WK"及び DO 110 における配列"FLUX"は，インデックス参照ではなく直接参照となっていることに注意されたい．これにより，メモリアクセスの効率化も同時に期待できる．

リスト 3.3 流束計算の変更例

```
    DO 100 IE = 1, Nedges
      HLLEW=FLUX_FUNC(arguments)
      EDG_WK(IE)=HLLEW
100 CONTINUE
    DO 110 N = 1, Nnode
      NEDGE = IEDGE_in_NODE(0,N)
      DO 111 IE=1,NEDGE
        IEDGE=IEDGE_in_NODE(IE,N)
        XSIGN=SIGN(1.0D0,IEDGE)
        IEDGE=ABS(IEDGE)
        FLUX(N)=FLUX(N)+XSIGN*EDG_WK(IEDGE)
111   CONTINUE
110 CONTINUE
```

4. 性能測定条件

性能測定は，三次元翼周りの粘性流解析により行った．測定のために設定した初期条件を表 4.1 に示す．性能測定のために設定した計算格子点の数を表 4.2 に示す．MPI による並列実行時のプロセス数は，スカラ版では 2 および 8 プロセス，スレッド版では 2 プロセス固定とした．このとき，分割された領域間で通信を行うために余分の格子点が付与される．ここでは，この通信のためにオーバーラップして設けられた格子も含めた格子点数を示した．実際に解析が行われる格子点数は「正味」として示した．なお，この程度のプロセス数であれば，ノード間通信のオーバーヘッドは無視できる．

表 4.1 性能測定に使用した主な初期条件

設定内容	設定値
時間積分ステップ数	500 ステップ
クーラン数	1.0×10^{5}
マッハ数	0.8
迎角	0 度
レイノルズ数	1.0×10^{6}
壁面条件	断熱壁
流れの種別	層流

表 4.2　性能測定に使用した計算格子点の数

要素名	要素			辺	格子点
	三角錐	三角柱	四角錐		
2 プロセス					
proc.0	716,199	0	0	861,171	128,802
proc.1	481,662	141,636	879	870,779	160,656
小　計	1,197,861	141,636	879	1,731,950	289,458
合　計			1,340,376	1,731,950	289,458
正　味					
小　計	1,173,840	141,636	879	1,690,955	280,969
合　計			1,316,355	1,690,955	280,969

5.　性能測定結果および評価

性能測定結果を以下に示す．各表中において，実行方法を（CPU 数ープロセス数ースレッド数）で表記した．純 MPI による並列化モデルでは，スレッド数を 0 と表記した．4 スレッド実行時には，IMPACT による自動スレッド並列化機能を用いた．

例 1：2-2-0
　　2CPU を使用し，2 プロセス，純 MPI で実行．
例 2：2-2-4
　　2CPU を使用し，2 プロセス，4 スレッド，
　　MPI と自動スレッド並列によるハイブリッド実行

また，測定は各所にタイマを挿入し，経過時間を測定することにより行った．以下では，統合スパコンのプロジェクトシステムによる測定結果を"新システム"として示すとともに，比較のため統合スパコン導入前に調布事業所に導入されていた大規模 SMP クラスタ富士通製 PRIMEPOWER HPC2500 による測定結果を"旧システム"として示す．

5.1　スカラ版測定結果

スカラ版の性能測定結果を表 5.1 および図 5.1 に示す．並列実行モデルは純 MPI である．ここで，実行開始から終了までに要した経過時間を「全体」として示すとともに，その内訳として右辺残差の計算（「右辺」），高次精度化で必要となる物理量の勾配の計算（「勾配」），Venkatakrishnan の制限関数の計算（「制限関数」）および LU-SGS 法による連立方程式の解の計算（「LU-SGS」）の経過時間を示し，それらの計算の合計時間を「小計」として示した．また，旧システムに対する新システムの性能向上率を表 5.2 に示す．性能向上率は，旧システムでの経過時間を新システムでの経過時間で除することによって求めた．

新システムにおける 1CPU，2 プロセス実行（2 コア使用）は，旧システムにおける 2CPU，2 プロセス実行に対して，3 倍弱の性能向上を示している．新システムの SPARC64 VII チップは，基本的に，旧システムの SPARC64 V チップ 4 個を 1CPU チップに集積するとともに，動作周波数を 1.3GHz から 2.5GHz に向上させたものであることから，同じコア数（旧システムは 1CPU1 コア）を使用した場合には，2 倍程度の性能向上が期待されるが，本性能測定ではそれを十分上回る 3 倍弱の性能向上が得られている．SPARC64 VII チップのコアは，SPARC64 V チップに対して演算性能の強化が図られており，システムとしてのメモリアクセス能力も向上しているので，その効果が現れたものと考えられる．

しかし，新システムにおける 2CPU，8 プロセス実行（8 コア使用）の場合，旧システムにおける 8CPU，8 プロセス実行に対して，性能向上率は 2.6 倍程度まで低下している．これは，チップ内の 4 つのコアが，キャッシュおよびメモリを共有していることによるアクセス競合に原因があると考えられる．現状では，マルチコア化による 1 チップあたりの演算能力の向上に対して，十分なデータの供給を行えるだけのメモリアクセス性能の向上は図られていない[3]．そのような中で，新旧システムの 8 プロセス同士の比較においても，クロック比 2 倍を十分上回る 2.5 倍以上の性能向上率が得られたことから，少なくとも旧システムとの比較の上では十分な性能であり，演算性能とメモリアクセス性能のバランスも取れていると言えるのではなかろうか．

表 5.1 スカラ版経過時間（単位：秒）

処　理	旧システム		新システム	
	2-2-0	8-8-0	1-2-0	2-8-0
全　体	5360.71	1364.21	1798.01	526.27
右　辺	857.97	195.26	314.35	77.72
勾　配	1572.96	397.72	588.43	163.38
制限関数	662.85	138.61	235.09	55.55
LU-SGS	1243.24	341.65	346.30	119.71
小　計	4337.02	1073.24	1484.17	416.36

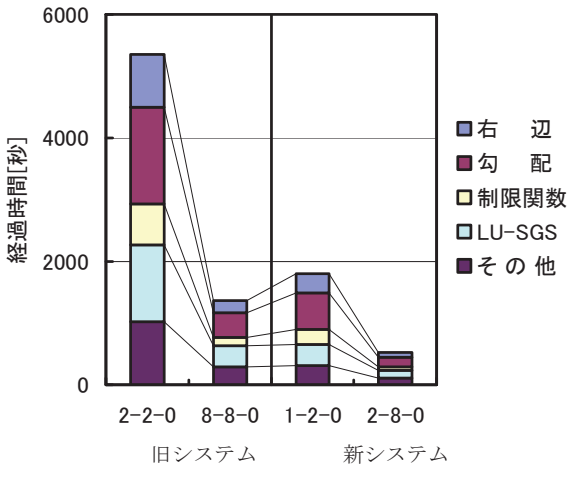


図 5.1 スカラ版経過時間

[3] 貫通電極を利用した CPU とメモリの積層実装[*10*]やシリコンフォトニクス技術によるメモリバンド幅の向上[*11*]など，種々の試みがなされているが，実用化には至っていない．

表 5.2 スカラ版性能向上率

処 理	旧システム 対 新システム		
	2-2-0 対 1-2-0	2-2-0 対 2-8-0	8-8-0 対 2-8-0
全 体	2.98	10.19	2.59
右 辺	2.73	11.04	2.51
勾 配	2.67	9.63	2.43
制限関数	2.82	11.93	2.50
LU-SGS	3.59	10.39	2.85
小 計	2.92	10.42	2.58

表 5.3 に新旧システムにおけるスケーラビリティを示す．スケーラビリティは，それぞれのシステムにおける 2 プロセス実行の経過時間を，8 プロセス実行の経過時間で除すことにより求めた．従って，理論的には 4 となるべきものである．

旧システムにおいては，ほぼ理論値に近いスケーラビリティが得られているのに対し，新システムでは，旧システムと比較すると低いスケーラビリティに留まっており，マルチコアにおけるメモリアクセスの厳しさが伺える．とは言え，全般に理論値 4 の 8 割以上のスケーラビリティが得られており，特に性能が不十分というわけではない．ただし，LU-SGS 法の計算部分については，約 2.9 倍と 3 倍を切るスケーラビリティしか得られていない．この部分は，旧システムにおいても他の部分との比較においては低いスケーラビリティしか得られていないため，並列化手法に性能低下の原因があると考えられる．性能改善が可能かどうかも含めて，具体的な原因を究明することは今後の課題である．

表 5.3 スカラ版スケーラビリティ

処 理	旧システム	新システム
	2-2-0 対 8-8-0	1-2-0 対 2-8-0
全 体	3.93	3.42
右 辺	4.39	4.04
勾 配	3.95	3.60
制限関数	4.78	4.23
LU-SGS	3.64	2.89
小 計	4.04	3.56

5.2 スレッド版測定結果

スレッド版の性能測定結果を表 5.4 および図 5.2 に示す．内訳については，表 5.1 および図 5.1 と同様である．また，旧システムに対する新システムの性能向上率を表 5.5 に示す．性能向上率は，旧システムでの経過時間を新システムでの経過時間で除することによって求めた．並列実行モデルは IMPACT による自動スレッド並列と MPI 並列を組み合わせたハイブリッド実行である．

新旧システムの 1 スレッド同士の比較(2-2-1 対 1-2-1)では約 3 倍の高性能が得られているが，4 スレッド同士の比較（8-2-4 対 2-2-4）では，2 倍前後とほぼクロック比に近い性能向上率に低下し，旧システムの 1 スレッドに対する新システムの 4 スレッド実行（2-2-1 対 2-2-4）ではクロック比から期待される性能向上率 8 倍を下回ってしまっている．特に勾配の計算における性能向上率の低下が著しい．この原因については，ハードウエアの問題ではなく負荷の偏りによるものであることが考えられる．その確認および解決策については，「6. 分析と考察」において示す．

表 5.4 スレッド版経過時間（単位：秒）

処 理	旧システム		新システム	
	2-2-1	8-2-4	1-2-1	2-2-4
全 体	6012.07	1810.94	1982.88	804.28
右 辺	1036.89	273.57	373.58	122.80
勾 配	1660.64	454.79	588.09	241.32
制限関数	775.84	208.14	261.47	99.48
LU-SGS	1374.72	389.80	446.29	176.77
小 計	4848.09	1326.3	1669.43	640.37

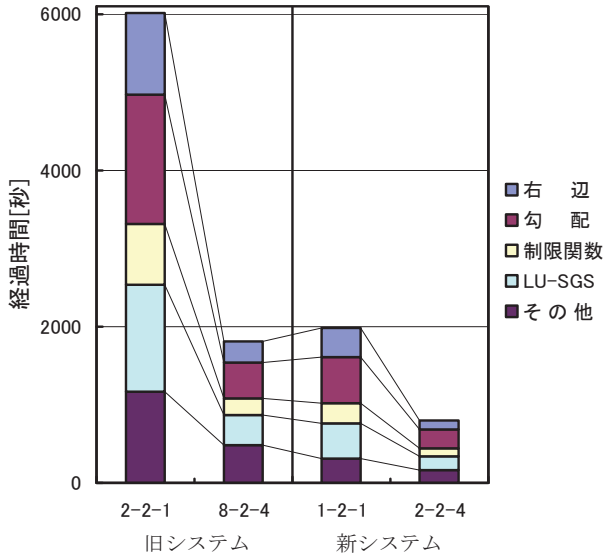


図 5.2 スレッド版経過時間

表 5.5 スレッド版性能向上率

処 理	旧システム 対 新システム		
	2-2-1 対 1-2-1	2-2-1 対 2-2-4	8-2-4 対 2-2-4
全 体	3.03	7.48	2.25
右 辺	2.78	8.44	2.23
勾 配	2.82	6.88	1.88
制限関数	2.97	7.80	2.09
LU-SGS	3.08	7.78	2.21
小 計	2.90	7.57	2.07

表 5.6 にスレッド版の新旧システムにおけるスケーラビリティを示す．スケーラビリティは，それぞれのシステムにおける 1 スレッド実行の経過時間を，4 スレッド実行の経過時間で除すことによって求めた．新システムにおけるスケーラビリティは，全体に理論値の 8 割を下回っており，十分な性能が得られていない．このことに

対する主な原因も，先に述べた性能向上率低下の原因と同一であると考えられる．

表 5.6 スレッド版スケーラビリティ

処　理	旧システム 2-2-1 対 8-2-4	新システム 1-2-1 対 2-2-4
全　体	3.32	2.47
右　辺	3.79	3.04
勾　配	3.65	2.44
制限関数	3.73	2.63
LU-SGS	3.53	2.52
小　計	3.66	2.61

6. 分析と考察

6.1 スレッド版の性能低下の要因

スレッド版の性能低下の一因として，各スレッドの負荷の不均衡が考えられる．そこで，OpenMP のディレクティブを挿入し，処理を動的に割り当てることにより負荷を均衡させると性能が改善するかどうかを調べた．

リスト 6.1 に処理の動的割り当て（dynamic）指定を行うディレクティブの挿入例を示す．この例では，各スレッドは，DO ループのインデックスを 1000 ずつ処理する．そして，割り当てられた処理が終了したスレッドから，順次，まだ処理されていないインデックスを処理する．以下，スレッド版の主要なループに OpenMP のディレクティブを挿入したコードを OMP 版と呼ぶ．

リスト 6.1　OpenMPディレクティブ挿入例

```
!$OMP PARALLEL PRIVATE(IE,N,...)
!$OMP DO SCHEDULE(DYNAMIC,1000)
      DO 100 IE = 1, Nedges
         HLLEW=FLUX_FUNC(arguments)
         EDG_WK(IE)=HLLEW
  100 CONTINUE
!$OMP END DO
!$OMP DO SCHEDULE(DYNAMIC,1000)
      DO 110 N = 1, Nnode
         NEDGE = IEDGE_in_NODE(0,N)
         DO 111 IE=1,NEDGE
            IEDGE=IEDGE_in_NODE(IE,N)
            XSIGN=SIGN(1.0D0,IEDGE)
            IEDGE=ABS(IEDGE)
            FLUX(N)=FLUX(N)+XSIGN*EDG_WK(IEDGE)
 111    CONTINUE
 110 CONTINUE
!$OMP END DO
!$OMP END PARALEL
```

OMP 版について，新システムで 2CPU 2 プロセス 4 スレッド実行（2-2-4）の計算時間を測定した結果を表 6.1 および図 6,1 に示す．表 6.1 には，同じコア数を使用して実行したスカラ版（2CPU8 プロセス；2-8-0）とスレッド版（2CPU 2 プロセス 4 スレッド；2-2-4）の測定結果も再掲した．経過時間は，スカラ版 ＜OMP 版 ＜ スレッド版の順となった．表 6.2 には，スカラ版と OMP 版の性能比（「スカラ版対 OMP 版」）および OMP 版とスレッド版の性能比（「OMP 版対スレッド版」）を示した．OpenMP ディレクティブの挿入により，スレッド版に比べて全体で 3 割，特に勾配の計算と制限関数の計算では，4 割を超える計算時間が短縮され，スレッド版において 5 割を超えていたスカラ版に対する性能差が 2 割を切るところまで縮小した．このことから，OpenMP を用いて負荷を動的に制御する方法がスレッド版に有効であることが確認できた．

表 6.1 OMP 版経過時間（単位：秒）

処　理	**OMP 版** **2-2-4**	スカラ版 2-8-0	スレッド版 2-2-4
全　体	**618.79**	526.27	804.28
右　辺	**90.92**	77.72	122.80
勾　配	**169.01**	163.38	241.32
制限関数	**66.88**	55.55	99.48
LU-SGS	**159.00**	119.71	176.77
小　計	**485.81**	416.36	640.37

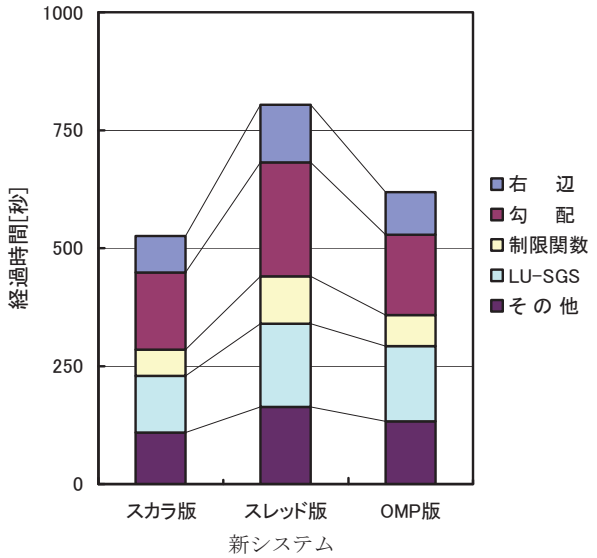


図 6.1 OMP 版経過時間

表 6.2 OMP 版とスカラ版，スレッド版の性能比

処　理	スカラ版対 OMP 版 2-8-0 対 2-2-4	OMP 版対スレッド版 2-2-4 対 2-2-4
全　体	1.18	1.30
右　辺	1.17	1.35
勾　配	1.03	1.43
制限関数	1.20	1.49
LU-SGS	1.33	1.11
小　計	1.49	1.32

以下，この性能向上の要因について考察する．一般に，スレッド並列化されたコードの実行時間 t は，

$$t = \bar{t} + \max_i(\delta t_i)$$

と表すことができる．ここに，右辺第一項$\bar{t}$は各スレッド

の実行時間の算術平均である．右辺第二項は負荷の不均衡を表す量であり，スレッド i の実行時間 t_i と平均実行時間 $\bar{t}$ の差 δt_i

$$\delta t_i = t_i - \bar{t}$$

の最大値である．OMP 版を上付き添え字 O によって，スレッド版を T によって区別するものとすれば，その実行時間の差 Δt は，

$$\begin{aligned}\Delta t &= t^T - t^O \\ &= \left(\bar{t}^T - \bar{t}^O\right) + \left[\max_i\left(\delta t_i^T\right) - \max_i\left(\delta t_i^O\right)\right] \\ &= \overline{T} + \delta t\end{aligned}$$

で与えられる．ただし，

$$\overline{T} = \bar{t}^T - \bar{t}^O$$
$$\delta t = \max_i\left(\delta t_i^T\right) - \max_i\left(\delta t_i^O\right)$$

とおいた．ここで，各スレッドの実行時間 t_i はメーカー提供の動的解析ツールであるプロファイラにより測定することができる．したがって，$\bar{t}$ および δt_i を計算することができ，以上の式を評価することが可能である．その結果を表 6.3 に示す．表 6.1 より求まる Δt=185.49 と完全に一致しないのは測定誤差である．

表 6.3 スレッド版と OMP 版の実行時間差の内訳

	Δt	$\bar{T}$	δt
測定結果［秒］	183.36	107.48	75.87
比 ［—］	1.00	0.59	0.41

この結果から，OpenMP ディレクティブの挿入による負荷不均衡の改善による効果が 76 秒程度あることがわかる．しかし，その性能向上に占める割合は 4 割程度であり，それ以外の要因による性能向上が，それを上回る約 6 割を占めることもわかった．

そこで，まずスレッド版と OMP 版の平均計算時間の差 $\bar{T}$（～107 秒）について考える．この性能向上の真の要因を特定することは，一般には非常に困難であるが，ここでは，スレッド版において 1.112%であった L2 キャッシュ・ミス率（以下 L2 ミスと略す）が OMP 版で 0.840%に減少したことに着目し，このことが実行時間にどの程度の影響を与えたか推定することを試みる．L2 ミスの減少は，メモリと L2 キャッシュの間のデータ転送量の減少を意味するので，その結果，メモリアクセス時間が短縮される．L2 ミスとデータ転送量は比例関係にあり，単位時間あたりのデータ転送量であるメモリースループットが一定であると仮定すれば，L2 ミスとデータ転送時間も比例する．メモリアクセス時間 t_a は，実行時間 t に占めるメモリアクセスコストの割合を A_c〔%〕とすれば，

$$t_a = t \times A_c / 100$$

によって求めることができる．L2 ミス $L2$ とメモリアクセス時間 t_a は比例関係にあるのだから，L2 ミスが $L2_1$ から $L2_2$ に変化した場合に，メモリアクセス時間が t_{a1} から t_{a2} に変化したとすれば，

$$\frac{t_{a2}}{t_{a1}} = \frac{L2_2}{L2_1}$$

が成り立つ．よって，メモリアクセス時間の差 Δt_a は，

$$\Delta t_a = t_{a1} - t_{a2} = \left(1 - \frac{L2_2}{L2_1}\right) t_{a1}$$

として求めることができる．プロファイラによって A_c および $L2$ を実測することが可能であり，その結果からは Δt_a=100.13 秒が得られた．このことから，平均実行時間の差 $\bar{T}$（～107 秒）は，おそらく L2 ミスの減少によるものであると推定できる．

このような推定が可能なのは，スレッド版と OMP 版とで，DO ループの構造が全く同一であることに因ることに注意されたい．一般には，ソースプログラムの変更を伴うことにより，L2 ミスだけでなく演算性能も変化するので，このような簡単な方法で推定することは困難である．

一方で，スレッド版と OMP 版で，その DO ループの構造自体に差異はないにもかかわらず，なぜ L2 ミスが減少したか，その理由はよくわからない．スレッド版に比して OMP 版では，処理の単位が細分化されているので，メモリアクセスの不連続性がより大きく，L2 ミスが増大する傾向にあると考えるのが自然である．ひとつ考えられるのは，コア間の L2 キャッシュ共有がより有効に機能するようになった可能性である．スレッド版では，各スレッドはそれぞれが十分に多い要素または辺を番号順に計算していく．そのため，各スレッドは相互に遠く離れた解析空間上の領域を計算している可能性が高い．一方，OMP 版では，より小さい単位に分割されるので，相互のスレッドがより近接した点を計算している可能性が高くなる．そのため，あるスレッドが計算に使用したデータがキャッシュに残っている間に，他のスレッドがそのデータを計算に使用する可能性は，OMP 版の方が，スレッド版にくらべて高いと考えられる．

次に，負荷不均衡の改善について，さらに詳しく調べる．一般に，並列実行における負荷不均衡は演算が各スレッドに均等に割り当てられていないことに起因すると考えられる．しかし，非構造格子の場合，その性質上配列をランダムにアクセスする必要があるため，各スレッドにおける L2 ミスが均等ではなくなることにより，各スレッドごとのメモリアクセス時間に不均衡が生じる可能性も無視できないと考えられる．そこで，δt_i をメモリアクセス時間のその平均時間からの差 $\delta t_{a\,i}$ と演算時間のその平均時間からの差 $\delta t_{o\,i}$ に分けて

$$\delta t_i = \delta t_{ai} + \delta t_{oi}$$

として考える．スレッド i の実行時間に占めるメモリア

クセスコストの割合 A_{ci} からメモリアクセス時間 t_{ai} は，

$$t_{ai} = t_i \times A_{ci}/100$$

によって得られるので，δt_{ai} は，

$$\delta t_{ai} = t_{ai} - \bar{t}_a$$

として求められる．ただし，$\bar{t}_a$ はスレッド平均メモリアクセス時間である．また，演算時間の不均衡 δt_{oi} は

$$\delta t_{oi} = \delta t_i - \delta t_{ai}$$

として計算することができる．この測定結果を表 6.4，図 6.2 および表 6.5 に示す．表 6.5 において，定義より

$$\max_i(\delta t_i) = \max_i(\delta t_{ai} + \delta t_{oi}) \neq \max_i(\delta t_{ai}) + \max_i(\delta t_{oi})$$

であることに注意されたい．

表 6.4 スレッドごとの負荷不均衡　（単位[秒]）

	OMP 版			スレッド版		
	δt_i	δt_{ai}	δt_{oi}	δt_i	δt_{ai}	δt_{oi}
プロセス 0						
スレッド 0	103	82	21	147	112	35
スレッド 1	11	33	-22	97	119	-22
スレッド 2	7	31	-24	77	96	-19
スレッド 3	1	27	-27	-8	5	-12
プロセス 1						
スレッド 0	97	2	94	179	67	112
スレッド 1	-65	-53	-12	-1	6	-7
スレッド 2	-74	-59	-15	-170	-163	-7
スレッド 3	-80	-64	-16	-250	-241	-8

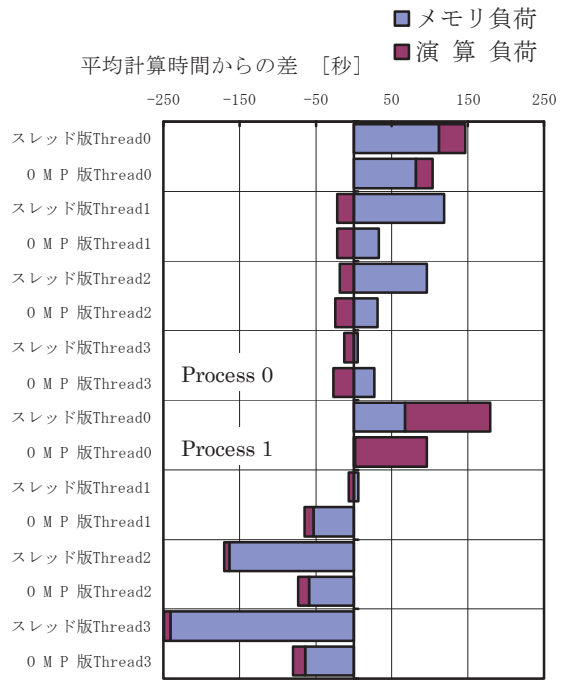


図 6.2 スレッドごとの負荷不均衡

表 6.5 負荷不均衡の改善　（単位[秒]）

	$\max_i(\delta t_i)$	$\max_i(\delta t_{ai})$	$\max_i(\delta t_{oi})$
スレッド版	179	119	112
OMP 版	103	82	94
Δt	76	37	18

この結果より，

a. プロセス 0 および 1 のいずれにおいてもマスタースレッド（スレッド 0）の負荷が高い
b. プロセス 1 のマスタースレッドに演算負荷が集中しており OMP 版においても解消されていない
c. プロセス 1 のマスタースレッド以外のスレッドのメモリ負荷が極めて低い

ことがわかる．特に a.からは，自動並列化されていない部分の存在が疑われる．並列化されていない場合には，動的な負荷分散を行っても効果が得られないことから，効果が限定的なものに留まったものと考えられる．並列化に関するスケーラビリティを低下させる原因となるものでもあり，プロセス 1 のマスタースレッドへの負荷集中に関しては，解消の可能性を含めてその原因を解明することが必要である．また，負荷不均衡の改善は，主にメモリアクセスの不均衡に対してなされたものであることもわかった．

以上，新システムにおける 4 スレッド実行による測定には，全て IMPACT と呼ばれる自動スレッド並列化オプションによって翻訳した実行可能ファイルを使用した．しかし，今回のアプリケーションに対しては，負荷のアンバランス等により，結果として自動並列化によるマルチコアの利用のみでは十分な並列化効果が得られず，OpenMP ディレクティブによる指示を必要とした．

6.2　新システムの利用指針と技術課題

スパコンを管理する計算センタとしては，広範なユーザに，より簡便な利用環境を提供するとともに，より効率の良い利用を促すことにより，限られた資源の有効な活用を図る必要がある．自動並列化は，より広範なユーザに簡便な利用環境を提供する上で極めて重要な役割を果たすことは明らかである．CPU メーカー各社が，マルチコア化によりチップあたりの性能向上を図っている現状では，少なくとも当分の間は，スパコンの性能向上もコア数を増やすことによってなされると予想される[4]．そのような中で CPU 内のコアを意識することなく，従来どおり 1CPU 1 コアであるかのように利用できる IMPACT のような自動並列・最適化技術の重要性は，今後益々増大して行くものと思われる．特に，ベクトル化されたプログラムと自動並列化機能の親和性に鑑みれば，過去の経緯からベクトル化されたプログラム資産を多く

[4]2009 年 5 月 13 日付読売新聞夕刊の報道によれば，富士通は 8 コアの次世代 SPARC64 チップ Venus（コード名）の開発に成功した．

保有するJAXAにおいて，その重要性は顕著であると言える．

しかし一方で，マルチコアCPUを効率的に利用することに関する問題点も明らかとなった．まず，現状の自動並列化では負荷の不均衡に対応出来ない，ということである．この問題点に関しては，OpenMPを併用することにより対処が可能であることがわかったが，このことは同時にCPUがマルチコアであることを意識した並列化を行わなければならないことを意味する．また，ノードにまたがる並列化には，分散メモリに対応可能なMPIに代表されるプロセス並列化手法を用いる必要があることから，複数の並列化手法を併用しなければならないということでもある．

加えて，再帰参照を如何に扱うかという問題もある．ベクトル計算機では，そもそも主要なループ内に再帰参照が含まれた段階で，性能が著しく低下することは避けられないので，再帰参照が含まれないようにコーディングしなければならない．しかし，このコーディングの制約は極めて厳しいものである．既に示したように非構造格子NSソルバで再帰参照を避けるためには「DOループの分割」や「色分け」などの手法を使う必要があるが，このような手法は，例えば，NSソルバの主要な適用分野であるCFDの知識とは全く関係のない，純粋に計算機利用技術に属するものである．多くの場合，スパコン利用者はCFDの解析を行うことを目的とした技術者や研究者であって，計算機の技術者ではないことを考えると，このような計算機利用技術に関する知識は，なるべく要求されないことが望ましい．また，スカラ計算機では，再帰参照を許した方がより効率的である場合があることは測定結果より明らかである．その意味では，ひとつのコアにひとつのプロセスを割り当てて実行する（純MPI並列モデル）ことも，現実的な選択肢のひとつである．この場合，コードにはMPI等を用いたプロセス並列化のみを行えばよいので，複数の並列化手法を用いる必要がない，という利点もある．一方で，このような方法で，CPU内のコア数の増大に対処して行けるのかという問題点もある．仮にノード数は現状と変わらずCPU内のコア数を増やすことによりスパコンの性能向上が図られた場合，コア内を自動並列化またはOpenMPによりスレッド並列化した場合には，コア数がいくら増えてもプロセス数は高々10^3の桁に留まる．しかし，プロセス並列のみで対応しようとした場合，コア数が100程度になればプロセス数は10^5となる．このような大規模なプロセス数になると，格子を各プロセスに割り当てるために分割する処理を実行することが極めて困難になると予想される．さらに，NS方程式の解法としてCFDで用いられている陰解法は，計算アルゴリズムが並列性を持たないため，プロセスごとに独立に適用される場合が多い．このような場合には，プロセス数が増大するとともに，時間発展の収束性が低下する懸念がある．このため，単にタイムステップ当たりの計算時間のみの優位性から1コア1プロセス実行を採用して良いのかという問題もある．プロセス間通信のオーバーヘッドについては，プロセス数が増加するとしてもCPU内のプロセス間通信がメモリアクセスによるものである限り，ノード間の通信に比べ十分早いことからそれほど問題にはならないと予想される．ただし，プロセス間の通信速度に大きな差がある場合，分割した格子空間をどのようにプロセスに割り当てるかによって性能が大きく異なる場合があるので注意を要する．

評価に使用したコードを，新システム上で実際に運用するにあたっては，以上を勘案した上でその運用方法を決定する必要がある．今回の測定では，ディスクアクセス性能や通信性能の問題が無視できるような条件で，主に演算性能とメモリアクセス性能の評価を行った．2008年11月に発表されたTOP500リスト[*12*]において第一位である米ロスアラモス研究所のRoadrunnerは、0.25B/FlopとFX1と比較して1/4程度のメモリバンド幅しか持たない中で，性能を出すためにアプリケーションの最適化に関してメモリバンド幅を超えないアルゴリズムを開発することまで行っている[*13*]．JAXA統合スパコンにおいては，メモリアクセス能力が高いため，このようなハードウエアを意識したアプリケーション開発が必要な状態にまでは至っていない．しかし，効率的な利用を考えたときには，ここまで見て来たように，種々の条件を勘案しなければならないことが明らかになった．今後，ますます演算性能に対して相対的にメモリアクセス能力が低下して行くことが予想される状況において，ハードウエアをより強く意識した利用技術を開発し，ノウハウを蓄積していくことが必要になると考えられる．

7. まとめ

JAXA統合スパコンの性能について，非構造格子NSソルバを用いて，主に演算性能およびメモリアクセス能力について評価を行った．その結果，新システムは，IMPACT並列実行により，旧システムと比較して十分高い性能を有することが確認できた．一方で，マルチコアCPUを有効に活用するためには，複数の並列化手法を複合的に適用しなければならない可能性や，再帰参照を含むコーディングを許容するか否かなど，考慮しなければならない点のあることもわかった．今後，よりハードウエアの構造を意識した利用技術の開発が必要になると考えられる．

8. 謝辞

本報告の性能評価および結果の分析においては，富士通株式会社より協力をいただいた．ここに謝意を表す．

付録 A

本付録においては，JAXA 統合スパコンの一部である共有メモリサブシステム（A システム）の性能評価結果について示す．表 A.1 および図 A.1 に，4 コアモデルの富士通製 SPARC Enterprise（SE） M9000 によるスカラ版の性能測定結果を示す. SE M9000 と FX1 とは CPU に同じ SPARC64 VII チップを使用している．しかし，FX1 が 1 ノード 1 CPU の分散メモリシステムであるのに対して，SE M9000 は大規模共有メモリシステムであり，メモリアクセス機構に大きな違いがある．また，表 A.2 に SE M9000 と FX1 の性能比を示す．ここでは，SE M9000 による経過時間を同じプロセス数で実行した FX1 の経過時間で除した値を示した．測定環境の問題から，SE M9000 を使用した計算全体に亘る経過時間の測定は実施できなかったため空欄とした．測定の結果，SE M9000 と比較して同じ CPU を使用した FX1 において，主要な処理全てにおいて 4 割以上，特に勾配の計算および LU-SGS 法の計算においては，2 倍以上の性能が出ており，FX1 のメモリアクセス機構の能力の高さが明らかとなった．

表 A.1 スカラ版経過時間（SE M9000，単位：秒）

処　理	1-2-0	2-8-0
全　体	----------	----------
右　辺	439.58	111.50
勾　配	1188.42	327.40
制限関数	332.60	79.10
LU-SGS	836.43	270.10
小　計	2797.03	788.1

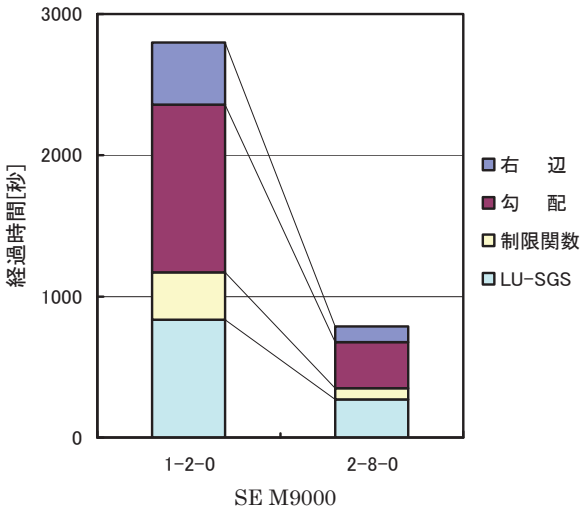


図 A.1 スカラ版経過時間（SE M9000）

表 A.2 スカラ版の性能比（FX1 対 SE M9000）

処　理	FX1 対 SE M9000	
	1-2-0	2-8-0
全　体	----------	----------
右　辺	1.40	1.43
勾　配	2.02	2.00
制限関数	1.41	1.42
LU-SGS	2.42	2.26
小　計	1.88	1.89

表 A.3 および図 A.2 に SE M9000 によるスレッド版の性能測定結果を示す．また，表 A.4 に SE M9000 と FX1 の性能比を示す．スレッド版においても，主要な処理のほぼ全てにおいて 4 割以上高い性能が出ており，FX1 に高いメモリアクセス能力があることがわかる．スレッド版において，スカラ版より性能比が低下する原因としては，既に明らかになった演算負荷の偏りの問題があげられる. 演算負荷の偏りは, 並列化効率の低下をもたらし，結果として演算性能を低下させる．この性能低下は，メモリアクセス機構の能力とは無関係であるので，メモリアクセス能力が向上しても性能の向上には結びつかない. 結果として，スカラ版に比べると低い性能比にとどまっているものと考えられる．とはいえ，2CPU 2 プロセス 4 スレッド実行（2-2-4）において 2 倍近い性能が得られていることから, SE M9000 に比して FX1 の性能は十分高いと言える．

表 A.3 スレッド版経過時間（SE M9000，単位：秒）

処　理	1-2-1	2-2-4
全　体	3160.31	1456.04
右　辺	503.96	208.70
勾　配	1052.30	456.42
制限関数	383.37	184.71
LU-SGS	872.75	346.66
小　計	2812.38	1196.49

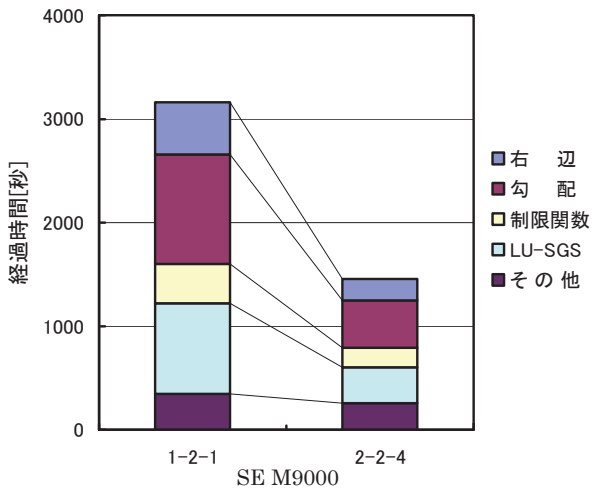


図 A.2 スレッド版経過時間（SE M9000）

表 A.4 スレッド版の性能比（FX1 対 SE M9000）

処 理	FX1 対 SE M9000	
	1-2-1	2-2-4
全 体	1.59	1.81
右 辺	1.35	1.70
勾 配	1.79	1.89
制限関数	1.47	1.86
LU-SGS	1.96	1.96
小 計	1.68	1.87

参考文献

[*1*] InfiniBand Trade Association
http://www.infinibandta.org/home

[*2*] Message Passing Interface Forum
http://www.mpi-forum.org/

[*3*] OpenMP.org
http://openmp.org/wp/

[*4*] Obayashi, S. and Guruswamy, G. P., "Convergence Acceleration of a Navier-Stokes Solver for Efficient Static Aeroelastic Computation", AIAA Journal, Vol. 33, No. 6, pp.1134-1141, 1995.

[*5*] Venkatakrishnan V., "On the Accuracy of Limiters and Convergence to Steady State Solutions.", AIAA Paper, 93-0880, 1993.

[*6*] Sharov, D. and Nakahashi, K., "Reordering of 3-D Hybrid Unstructured Grids for Vectorized LU-SGS Navier-Stokes Computations", AIAA 97-2102, 1997.

[*7*] Sharov, D. and Nakahashi, K., "Reordering of 3-D Hybrid Unstructured Grids for Lower-Upper Symmetric Gauss-Seidel Computations", AIAA Journal, Vol. 36, No. 3, 1998

[*8*] Fujita, T, et. al, "Evaluation of Parallelized Unstructured-grid CFD for Aircraft Applications", Proc. of Parallel CFD 2002.

[*9*] Sharov, D. , Luo, H. and Baum. J. D., "Implementation of Unstructured Grid GMRES+LU-SGS Method on Shared-Memory, Cache-based parallel Computers.", AIAA 2000-0927, 2000.

[*10*] 傳田 精一，「3 次元チップ積層のためのシリコン貫通電極(TSV)の開発動向」，表面技術，Vol.58, p.712 (2007)

[*11*] 西 研一，「LSI オンチップ光配線技術」，STARC シンポジウム 2007
http://www.starc.jp/download/sympo2007/08_nishi.pdf

[*12*] TOP500 Supercomputing Sites
http://www.top500.org/

[*13*] 霜田 善道，「PowerXCell と線形計算」
九州大学情報基盤研究開発センター 先端的科学計算に関するフォーラム 2008
http://www.cc.kyushu-u.ac.jp/scp/users/forum2008-09/shimoda.pdf